UNITED STATES DISTRICT COURT

FOR THE NORTHERN DISTRICT OF CALIFORNIA

OAKLAND DIVISION

EPIC GAMES, INC., Plaintiff, Counter-defendant, v. APPLE INC., Defendant, Counterclaimant.	Case No. 4:20-cv-05640-YGR-TSH **REBUTTAL WRITTEN DIRECT TESTIMONY OF DR. WENKE LEE** The Honorable Yvonne Gonzalez Rogers Trial: May 3, 2021 **Ex. Expert 15**

[redacted] REDACTED VERSION OF DOCUMENT SOUGHT TO BE SEALED

I. Qualifications

1. I, Wenke Lee, am a Professor and John P. Imlay Jr. Chair of the College of Computing at the Georgia Institute of Technology, where I conduct research in systems and software security, network security, applied cryptography, and operating system design. I am also the Executive Director of the Institute for Information Security & Privacy and served as the Director of the Georgia Tech Information Security Center from 2012-2015.

2. I have a B.Sc. degree in Computer Science from Sun Yat-Sen University, a M.Sc. degree in Computer Science from The City College of New York and a Ph.D. in Computer Science from Columbia University.

3. I am a Fellow of the Association for Computing Machinery (ACM) and of the Institute of Electrical and Electronics Engineers (IEEE), a Member of the IEEE Computer Society, a recipient of the ACM Special Interest Group on Security, Audit, and Control Outstanding Innovation Award (2019), and a recipient of the NSF Career Award (2002).

4. I have studied systems and software security, network security and malware analysis for over 25 years. I have also performed extensive research in the area of mobile system security. My research includes analyzing both on-device and off-device security measures.

5. My research team and I first became interested in analyzing iOS security in 2012 because the closed-source nature of iOS presented challenges to such analysis. At the time, there was also marketing and press coverage suggesting that Apple's mandatory app review process ("App Review") made iOS very secure. My team and I determined to analyze security vulnerabilities in the App Review process.

6. This effort resulted in the publication of a 2013 paper called "Jekyll on iOS: When Benign Apps Become Evil", which demonstrated a novel way for malicious third-party apps to evade detection during App Review. Specifically, the technique allows an app to function differently after it has been downloaded onto a device than it functioned during App Review, without any modification of its code after submission to Apple. This type of behavior, often referred to as a "backdoor switch", is something that has been and continues to be difficult to detect as part of the App Review process. As a result, a key recommendation resulting from our Jekyll experiment was for Apple to improve on-device security monitoring mechanisms on iOS.

7. In 2014, my team and I authored a paper called "On the Feasibility of Large-Scale Infections of iOS Devices", which documented an exploitation of a vulnerability in the iTunes Sync process that covertly downloaded apps outside of the App Store. This vulnerability allowed attackers to distribute malicious Apple-signed apps (like a Jekyll app) and apps that have not gone through App Review on iOS devices while iPhones were physically connected to a compromised computer (such as for syncing or backing-up purposes). While this type of attack is less relevant in today's world, where syncing of data often occurs wirelessly, our work showed that contrary to common belief at the time, large-scale infections were feasible on iOS devices through physical connections to other computing devices.

II. Assignment

8. I have been retained by counsel for Epic Games, Inc. to, among other things, evaluate and respond to certain aspects of an expert report submitted on behalf of Apple Inc.'s technical expert, Dr. Aviel Rubin.

9. My opinions in this matter are based on specialized knowledge arising from my training, study, and experience in the fields of computer science and systems security. I have relied on public-facing Apple documents, peer-reviewed literature, newspaper articles and other sources that experts in my field typically and reasonably rely on in formulating their opinions. I have also relied on internal Apple documents and other case materials, including deposition testimony of Apple representatives.

III. Summary of Opinions

10. I disagree with Dr. Rubin's position that Apple's App Review and exclusive distribution model provide significant security benefits that could not be achieved if third-party app distribution were permitted. I further disagree with Dr. Rubin's position that enabling third-party app distribution will weaken security on iOS.

11. I have evaluated the iOS security model based on Apple's *own* stated goals and processes for enforcing security on iOS. My analysis led me to conclude that, contrary to Dr. Rubin's claims, the same security features Apple seeks to enforce on iOS can be achieved without the need for exclusive distribution. For instance, third parties could perform the same security screening steps taken during App Review, and Dr. Rubin does not dispute this. Third parties can also perform developer identity verification and code signing. Most importantly, all of the on-device mechanisms that enforce security on iOS are independent of the app distribution model. (*See pp. 4-12.*)

12. Contrary to Dr. Rubin's assertions, Apple can support a more open distribution model on iOS by implementing features and tools that it already employs on the related macOS today. These include default security features that enable users to download only apps that have been screened for malware and signed, and tools that quickly identify, block and remove malware from both signed and unsigned apps. (*See pp. 12-13.*)

13. While Dr. Rubin claims that mobile devices have unique security considerations, he fails to establish that any of those considerations necessitate all iOS apps being distributed exclusively through Apple's App Store. For example, Dr. Rubin's claim that iOS devices are susceptible to getting misplaced, lost or stolen ignores the fact that Apple has addressed this vulnerability through on-device features, such as Touch ID, Face ID and Find My iPhone protections, that are completely orthogonal to app distribution. (*See pp. 13-14.*)

14. Dr. Rubin's comparison of security between platforms is misleading and incomplete. *First*, his comparative analysis notably omits Apple's macOS—a platform designed and operated by the same company that uses the same kernel as iOS. *Second*, Dr. Rubin's quantitative analysis of security on iOS relative to Android is irrelevant because iOS does not need to adopt

the Android security model to enable third-party distribution. *Third*, the statistics that he uses in this analysis are unreliable and misleading. *Fourth*, his analysis of the Android China market ignores other factors affecting that ecosystem which make it an unreliable comparison. (*See pp. 14-16.*)

15. Dr. Rubin overstates the security benefits of App Review. App Review is limited in its ability to catch less obvious, sophisticated security attacks, particularly given Apple's prioritization of efficiency over accuracy with respect to its review process. App Review focuses on many *non*-security issues. (*See pp. 17-20.*)

16. Contrary to what Dr. Rubin suggests, eliminating exclusive app distribution would not lead to an experience that is less secure than what Apple already accepts on macOS today. Among other things, users could choose to download apps exclusively through Apple's App Store and continue to experience the same level of security as they do on iOS today. Moreover, the impact of malware downloaded by a user that chooses to download outside of the App Store will remain localized to the user's device due to Apple's on-device security features. Lastly, Dr. Rubin does not provide any support for his assertion that third parties would not have the incentive or resources to provide as secure of an App Store experience as Apple. (*See pp. 20-23.*)

17. Enabling a more open distribution model can improve security on iOS. For example, third-party distribution channels could implement review processes that are less burdened by the volume of apps distributed on their respective platforms. Third parties could also specialize in specific categories of apps and improve upon security screening methods that are uniquely tailored for certain types of apps. (*See p. 23.*)

18. Finally, the security and privacy benefits offered by Apple's In-App Purchase ("IAP") can also be offered by third-party payment solutions. Indeed, third-party payment systems already ensure secure and private payment transactions on iOS today. (*See pp. 24-26.*)

IV. The iOS Threat Model

19. Dr. Rubin asserts that "[a]ny evaluation of the security of iOS . . . must consider its context, objectives, potential attackers, and the manner in which users use their systems, or the iOS 'threat model'".[1] However, Dr. Rubin's discussion omits any record evidence of Apple's *own* threat modeling for iOS. The Apple materials I have reviewed do not suggest that Apple's decision to adopt exclusive app distribution on iOS was a result of threat modeling.

20. Apple announced in 2007 that it would permit distribution of third-party apps on the iPhone. Contemporaneous documents prepared by Apple's security team indicate that the iOS architecture would support third-party distribution outside of the App Store, and the question of whether to permit such distribution was perceived by Apple's security personnel as a "policy" decision, not a "security" one. A 2007 security white paper explained: "We will distribute third party applications through the iTunes Music Store. However, our model will allow for third

[1] Rubin Written Direct Testimony ¶ 22.

parties to distribute their own applications and for enterprise customers to deploy to their own devices."[2] Scott Forstall, former Senior Vice President of iPhone Software, who was responsible for security on the iOS platform, testified "[t]his is a technical document from a technical team who is building the security infrastructure, and so their statement here is that the model – the technical infrastructure they're building will allow for other distribution mechanisms."[3] The white paper explained, "*it's left as a policy decision* as to whether Apple signed applications are posted to the online store, or we allow developers to distribute on their own."[4]

21. In my review of the iOS security model, I focused on Apple's *own* stated goals and processes for enforcing security on iOS. The various on-device security protections Apple has established for iOS devices and the five security properties that Apple seeks to enforce through its App Review process are described in the written testimony of Dr. James Mickens.[5] I reviewed internal Apple data and documentation concerning Apple's App Review process and identified six steps that App Review performs to screen for the five security properties identified by Dr. Mickens. (*See* paragraphs 36-53.) I concluded, consistent with the internal Apple materials cited above, that the same security properties Apple seeks to enforce on iOS can be achieved without the need for exclusive distribution, just as they are achieved on macOS.

22. Dr. Rubin claims that Dr. Mickens and I have taken a narrow view of security that excludes privacy, trustworthiness, reliability and legal compliance. This is incorrect. Dr. Mickens discusses issues related to user privacy and legal compliance on the iOS platform,[6] and I will discuss reliability in the context of exploit resistance below (*see* Section V.A.iii(6)). I consider "trust" in terms of the technical definition of Trustworthy Computing:[7] computing that is available (*see* Sections V.A.iii(2), V.A.iii(6), and VIII.B), reliable (See Sections V.A.iii(6) and VIII.B), and secure (see Section V.A.iii). Moreover, I have evaluated the App Review steps that enforce both security and non-security properties (including topics such as quality assurance, content moderation and commercial compliance). I have concluded, and Dr. Rubin does not dispute, that both the security *and* non-security App Review steps can be performed by third parties. Finally, I note that Apple's approach to enforcing properties such as "trustworthiness" on iOS is inconsistent as it does not address what Dr. Rubin refers to as "social engineering" campaigns such as phishing that target iOS users through other crucial parts of its platform, including Safari, email and iMessage.

V. The Security Features that Apple Implements on iOS Can Also Be Enforced By Third Parties.

23. Dr. Rubin suggests Apple alone can implement the security protections that it has identified for the iOS platform. I disagree. I believe the security protections Apple currently

---

[2] PX877.3.

[3] Forstall Dep. 130:15-19.

[4] PX877.3 (emphasis added).

[5] Written Direct Testimony of James W. Mickens, Ph.D. ("Mickens Testimony") at pp. 5-12, 15-16, 19-25.

[6] *Id.* at pp. 22-25.

[7] https://www.wired.com/2002/01/bill-gates-trustworthy-computing/.

implements on iOS, except the on-device mechanisms, can be enforced by third parties. Apple's on-device security protections do not require exclusive distribution.

A. Dr. Rubin Does Not Contest that Third Parties Can Perform the Same Security Screening Steps as App Review.

24. Dr. Rubin claims that the App Review process provides significant security protections. Even assuming that to be true, Dr. Rubin does not dispute that third parties could perform the same security screening steps taken during App Review.

25. App Review screens for security using some combination of manual and automated reviews. Both techniques are well understood and can be performed by third parties.

i. Manual review

26. Apple relies on two groups of human reviewers to screen for security violations during App Review: a group of standard App Reviewers, who test each app's basic functionality for not more than a few minutes, and a group of more sophisticated reviewers on the Technical Investigation team.

27. Based on my review of internal Apple data and documents, App Reviewers do not appear to possess expertise that a third party could not develop or acquire. App Reviewers are not required to have sophisticated technical backgrounds and Apple does not appear to make hiring decisions on this basis. According to Philip Shoemaker, former Senior Director of App Store Review, the relevant qualifications were "that [applicants] understood how to use a Mac, that they understood how to use an iPhone, that they understood a little about the Apple brand . . . [that they] could breathe, they could think, and typically they came from the Apple [G]enius stores".[8] Apple "geniuses" are employees at Apple retail stores who assist customers with technical support needs; they are hired based on "an aptitude for acquiring skills in technical repairs and an eagerness to learn about all Apple products and devices".[9]

28. If App Reviewers detect unusual activity, they can escalate the app for a Technical Investigation by a different team within Apple. Apple requires candidates to have some additional technical qualifications to join its Technical Investigation team. For example, senior positions within the Technical Investigation team require computer language proficiency and app development experience.[10] However, none of these qualifications suggest capabilities that could not be obtained by third parties.

ii. Automated review

29. The primary tools that Apple uses to perform its automated review of security properties are static and dynamic analyzers. Dr. Rubin has previously described static and dynamic

[8] Shoemaker Dep. 38:1-7; 35:24-36:3.
[9] https://jobs.apple.com/en-us/details/114438151/us-genius.
[10] PX2118.1.

analyses as "common techniques used to analyze software for malware or other undesirable behaviors".[11] I agree.

30. Static analysis is a well-understood method that examines the code without running the program. The process entails understanding the code structure and helping to ensure that the code adheres to certain standards and rules. Oftentimes, static analyzers rely on heuristics to flag possible violations of programming code. Apple uses static analyzers to screen for private Application Program Interfaces ("APIs") and malware.

31. There are many existing static analyzers that third parties can use to screen for private APIs and malware. Apple itself has acquired third-party static analysis tools. For instance, in 2015, SourceDNA, an app analytics company that built static and dynamic analyzers, identified a data breach on iOS devices that resulted from malware evading detection during App Review; Apple acquired the company in 2016.[12] Apple continues to use publicly available static analysis tools, including a third-party static analyzer called Coverity, and has explored licensing other public tools like Semmle.[13] Third-party static analyzers, like Appthority and Mobile Security Framework ("MobSF"), could offer similar functionality to Apple's own static analysis tools, particularly in identifying private APIs and malware in apps. There are also a number of open-source tools for reviewing apps that support static analysis of iOS applications.

32. In addition to using static analyzers, Apple uses dynamic analyzers to screen for private APIs. Dynamic analyzers provide insight into an app's behavior while the app is running. They can catch security violations or problematic behaviors that are not easily detected through static analysis, including obfuscated calls to private APIs. The Technical Investigation team also uses dynamic analyzers to take a deeper look into possible guideline violations that are not caught by human reviewers or static analyzers.[14]

33. Dynamic analyzers are not an Apple-specific technology. Third parties, such as Hopper and Hex Rays, have built dynamic analyzers that are widely available for use. Apple's internal teams have previously used third-party tools as part of the App Review process.[15,16] Apple's use of third-party dynamic analyzers supports my conclusion that third parties can screen for private APIs and malware using static and dynamic analyzers available to the public.

34. Dr. Rubin asserts that Apple uses proprietary machine learning tools during App Review. While he does not provide any specifics about these tools, machine learning has been used by third parties for malware analysis and could be used by third-party app review processes.

---

[11] DX4885 at ¶ 73.
[12] Kosmynka Dep. 127:23-25.
[13] PX2350.1.
[14] PX2174.10; PX2090.1.
[15] PX2174.9.
[16] PX2090.1.

iii. App Review's security screening steps can be performed by third parties.

35. Mr. Shoemaker testified that every step in Apple's App Review process is susceptible to replication by another app store and agrees that implementing an equivalent review process "[is] just a matter of investing the money and taking the time and thinking about the problem and addressing it".[17] [redacted]

36. This is consistent with my analysis. Through my review of Apple's internal documentation and deposition testimony, I have derived six security-related properties that Apple screens for through its manual and automated review processes: 1) software development kit ("SDK") version compliance, 2) private API usage, 3) malware, 4) sandbox compliance and entitlements, 5) user consent for private data access, and 6) bandwidth usage. I conclude that each of these can be performed by third parties.

(1) SDK Compliance

37. Apple ensures that apps meet Apple's standards regarding the use of up-to-date SDK versions.

38. Third parties can screen apps to ensure that they are using the most up-to-date SDK versions. One way in which third parties can screen for SDK compliance is to unzip the contents of the app's package to examine the app's property list files. One of these property list files will reveal the SDK version that the app was built with. A third party could easily cross-reference this information with Apple's requirements for SDK compliance, which Apple publishes on its website. It would be simple and straightforward for third parties to write an automated script to perform this step.

(2) Private API Usage

39. Apple uses a combination of manual review and automated tools to screen for usage of private APIs. While unauthorized use of private APIs can implicate iOS security, not all instances of third-party usage of private APIs raise security concerns. Instead, one common concern with private API usage is that apps may call upon APIs that have been deprecated (and made private)—apps that use these deprecated APIs may crash and provide customers with a bad experience.[19]

40. Third parties can also screen for private API usage. The first step to screening for private APIs is developing and maintaining a list of iOS private APIs. Apple does not publicize a list of

[17] Shoemaker Dep. 204:2-10.
[18] [redacted]
[19] Federighi Dep. 180:9-16.

its private APIs. However, third parties can use and have used tools to extract and maintain their own database of private APIs. For example, tools like RuntimeBrowser, IDA Pro and DyldExtractor can extract private APIs using a number of different techniques. One technique identifies private APIs by running an app and generating a list of all APIs (public and private) that are accessible to the app while the app is in use. From this comprehensive list, a third party can subtract out public APIs to arrive at a list of private APIs that the app should not be using.

41. This is further supported by the fact that Apple itself has "found over the decades that people will find things like private APIs" and that "informal researchers and enthusiasts in the past discovered private APIs and posted them to public forums or other distribution areas".[20] Once a third party has developed a list of private APIs, it can then use static and dynamic analyzers to screen for private API usage.

42. Contrary to Dr. Rubin's claim, third-party screening for private API usage does not require Apple to share its list of private APIs with third-party app stores and/or app developers. However, I do not believe anything would prevent Apple from sharing such information with trusted third parties (presumably subject to confidentiality agreements) if Apple believed such an arrangement could enhance security on the iOS platform.

(3) Malware

43. During App Review, human reviewers screen for malware, which are pieces of software that may, without the user's permission, gain access to information or attempt to run code unbeknownst to the user. C.K. Haun, Senior Director of Developer of Technical Services at Apple, points out that some of the inspection consists of "common-sense" checks: for instance, recognizing that "a simple Tic-Tac-Toe application typically does not need network access".[21] Third parties are also capable of hiring qualified reviewers to perform this type of function.

44. Other than common-sense checks, Apple utilizes automated tools to screen for malware. For example, Apple uses static analyzers to screen for keywords and code signatures that are associated with the presence of malware. Instances of previous iOS malware are well-known—many have been thoroughly studied and documented by third-party researchers. Third parties can leverage this research or perform their own analyses to identify malware signatures to run through their static analyzers.

45. Apple has also occasionally leveraged popular third-party malware analysis tools, such as Norton iAntivirus and ClamXAV, to detect malware in app submissions.[22] Mr. Shoemaker noted that before his departure from Apple in 2016, Apple used these tools to "[scan] the binaries for anything inside of the .zip file, [including] viruses".[23] While it is unclear whether Apple still relies on these third-party tools, third-party reviewers are just as qualified as Apple's own

[20] Haun Dep. 191:23-192:8.
[21] *Id.* 203:11-18.
[22] PX2106.2.
[23] Shoemaker Dep. 453:17-454:4.

reviewers to run commercially available anti-malware software as part of its malware screening process.

(4) Sandbox Compliance and Entitlements

46. All third-party apps are sandboxed on iOS, which means that each app is installed in its own "container" and is prevented from collecting or modifying information that is associated with other apps.[24]

47. The iOS operating system is predominantly responsible for enforcing sandboxing for all third-party apps. However, if there is suspicion that an app has maliciously escaped its sandbox, Apple will escalate the app for further investigation by the Technical Investigation team.

48. To screen for sandbox compliance, Apple uses a program called Aquarium to run the app and detect whether there are any changes to information and data outside of the app's sandbox, such as the contacts or photos on a device. This scanning process is not unique or proprietary—third parties can perform a similar screening process for sandbox compliance by scanning a device for differences before and after running an app.

49. Dr. Rubin states that App Review checks to make sure that an app is not requesting unnecessary entitlements, such as flagging when a Tic-Tac-Toe app seeks to act as spyware via access to an iPhone's microphone and camera. However, as noted above, Apple's human reviewers do not appear to have unique qualifications that would allow them to perform this step—third parties are capable of making decisions about entitlements and their relevance to the purported purpose of an app.

(5) User Consent for Private Data Access

50. Dr. Rubin provides very little detail concerning the precise steps Apple takes to protect privacy on iOS and why such steps require exclusive app distribution. Apple internal documents reveal that App Reviewers sometimes review documentation that developers provide with their app submission, including the app's privacy policies.[25] These policies may provide Apple a sense of what data the app intends to collect and how it plans to use the information. Third parties could review the same documentation.

51. In addition to the manual review of privacy policies, a 2013 email from Mr. Shoemaker shows that Apple considered building a privacy proxy tool, which would alert a reviewer when "a UDID [(unique device identifier)], or Mac address, or contacts are being sent" by an app over the internet, so that the reviewer could determine whether the app had permission to access that data.[26] It is unclear from Apple's documents whether Apple ever implemented this tool; even

[24] *See* Mickens Written Testimony at pp. 19-20.
[25] Haun Dep. 44:10-12.
[26] PX146.2.

assuming that Apple has employed such a tool, third parties are capable of building similar privacy proxies to detect for sensitive user information accessed by an app.

(6) Bandwidth Usage

52. As noted above, some reliability issues can implicate security. One such example is excessive bandwidth usage—malicious actors may create attacks using excessive bandwidth to consume all available bandwidth on a device, such that the device cannot receive legitimate network traffic, or in other cases, can cause the phone to crash.

53. As of 2010, Apple performed a bandwidth test to ensure that an app did not consume too much data (more than 4.5 MB of data during five minutes of testing).[27] Third parties can perform this test using the same built-in iOS functions that Apple's App Reviewers used to screen for bandwidth usage.

iv. Third-party stores and reviewers can communicate with Apple regarding iOS security.

54. Dr. Rubin asserts that Apple has an advantage over third-party review initiatives because of Apple's internal learnings concerning iOS and its ability to communicate internally among functional teams. For example, Dr. Rubin states that App Reviewers can make suggestions to the App Store Review Guidelines and to escalate app review findings and issues to other Apple teams through internal Apple channels. He suggests third parties cannot perform App Review because third-party reviewers would be incapable of utilizing Apple's open channels of communication that are only available to Apple Reviewers. I disagree.

55. Third-party reviewers could communicate with their own internal channels to make suggestions for their own guidelines or escalate their findings related to apps that are distributed on their stores.

56. Dr. Rubin questions whether or how each app store could establish a line of communication with Apple. Establishing such communications would not be burdensome or overly complicated. Apple already solicits information from the security community concerning vulnerabilities on iOS through its Security Bounty Program. Furthermore, in my experience, such knowledge accumulation and information sharing is quite common within the security community. Enabling this information sharing with a broader group of trusted industry participants (as opposed to one that is centralized within Apple only) could enable the security community to collaborate and improve upon iOS security.

[27] PX101.87.

B. Developer Verification and Code Signing Mechanisms Can Be Implemented by Third Parties.

57. As Dr. Mickens notes, existing iOS security properties also depend in part on developer verification and code signing. These, too, can be implemented by third parties without necessitating exclusive distribution through Apple's App Store.

58. Due to Apple's restrictions on third-party app distribution on iOS, Apple is currently the only company that verifies developers' identities for commercially available iOS apps. However, third parties have the capability to verify developers' identities, and they already do so for apps used on Android and Windows.

59. Several third parties already specialize in developer verification processes that could be used by or for third-party distribution channels on iOS. For example, DigiCert, a trusted certificate authority, requires its customers to provide their legal name, organization name, phone number, and address before requesting a code signing certificate. It then uses this information to "verify that the company requesting the [Secure Sockets Layer ("SSL")] certificate is in good standing… [and] that the individual requesting a certificate does, in fact, have authority to request a certificate for the domain in question".

60. Apple requires all third-party apps to be signed using a certificate obtained from Apple. Code signing is intended to help protect users against downloading and running software from disreputable developers. Among other things, code signing guarantees that a bad app can be traced back to a specific developer and that if the contents of the app have been altered after the app was signed, the app will not run on a device. Further, when malware is introduced to iOS, code signatures allow Apple to determine which developer is responsible for the security lapse and prevent that developer from distributing additional malware.

61. Code signatures are commonly used tools and could continue to be used on iOS, even if Apple allowed additional distribution channels. For example, for website verification purposes, web developers often sign their websites with SSL certificates from reputable third-party certificate authorities, like DigiCert or Verisign. Apple's browser verifies these SSL certificates on both iOS and macOS devices without the need for a signature authorized by Apple. Similarly, Microsoft allows third-party developers to sign their native Windows drivers with certificates from trusted third-party authorities. These examples demonstrate that it is technically feasible to track developer certificates, even those provided by third-party certificate authorities, in distribution models that are more open than the iOS model. Furthermore, Apple could also control which third-party certificate vendors it would accept for app-signing on the iOS platform.

C. On-Device Security Protections Are Independent of App Distribution.

62. In his written direct testimony, Dr. Mickens described on-device security mechanisms that enforce security on iOS devices.[28] All of the on-device security mechanisms on iOS devices

[28] Mickens Written Testimony at pp. 5-11.

are independent of the app distribution model. Put differently, enabling third party distribution on iOS has no impact whatsoever on the extent to which these features protect iOS devices.

VI. The Security Features that Apple Implements on iOS Can Be Enforced Using Tools that Apple Employs on macOS.

63. In his written testimony, Dr. Mickens describes the ways in which Apple has enabled a more open distribution model on Macs.[29] [redacted] This suggests that security on a device can be achieved without an exclusive app distribution model.

A. Apple Can Implement macOS Security Features that Enable Users To Download Only Apps that Have Been Screened for Malware and Signed.

64. To support a more open distribution model on macOS, Apple has implemented a number of additional security features to ensure security for apps distributed outside of the Mac App Store. The default security setting allows users to download only apps associated with developers whose identities have been verified by Apple and whose apps have been screened for malware and signed with an Apple-signed certificate. Thus, Apple implements the developer identification and code signature aspects of the security model without requiring all apps to be distributed through the Mac App Store.

65. Apple's "notarization" process screens for malware and other security issues in apps distributed outside of the Mac App Store. Apple's notarization process "is an automated system that scans [developers'] software for malicious content [and] checks for code-signing issues".[31] In certain cases, Apple's notary service also subjects an app to a manual review for security issues.[32] Moreover, Apple has the ability to issue a revocation ticket for malware, even if these apps were previously notarized.[33] If Apple discovers that a notarized app in fact contains malware, Apple can prevent it from being run on macOS devices. macOS regularly checks for revocation tickets to quickly block malware on the platform.[34]

66. Apple has also implemented software called Gatekeeper on macOS devices. Gatekeeper checks the app signature to "verify that the software is from an identified developer and that it has not been altered"[35] and also whether an app has been notarized by Apple. If Apple were to introduce a technology like Gatekeeper on iOS, Apple could similarly establish a default by which users could only run apps that have been screened for security issues and have been signed or notarized by Apple. Further, there is no technical reason why this Gatekeeper software could

---

[29] *Id.* at pp. 27-29.

[30] [redacted].

[31] https://developer.apple.com/documentation/xcode/notarizing_macos_software_before_distribution.

[32] Haun Dep. 223:23–224:2.

[33] https://support.apple.com/et-ee/guide/security/sec469d47bd8/web.

[34] *Id.*

[35] https://support.apple.com/en-us/HT202491

verify only Apple signatures and notary tickets. Internet browsers already check for signatures from multiple certificate authorities to encrypt traffic and verify a web server's identity. [36] As I discussed in Section V, third parties can screen for security issues; it would be technically straightforward for Gatekeeper to check for signatures and notary tickets from Apple-trusted third parties who can similarly vouch that a specific app has been screened for malware.

67. [redacted] XProtect screens both signed and unsigned apps against a database of signatures that indicate the presence of malware. [redacted]

68. [redacted] Both XProtect and the MRT would provide a mechanism to "turn off the spigot" on any malware on iOS without requiring all apps to be distributed through Apple's centralized App Store. Even if Apple does not choose to implement these two features on iOS, it could still use its control over the operating system to remove malicious apps that are distributed outside of the App Store.

69. Implementing all of these features on iOS would expand on iOS's on-device security protections and allow Apple to enable third-party app distribution while providing users with the ability to only download apps that have been screened for malware and signed. Mr. Federighi acknowledged that it would be technically feasible for Apple to implement these features on iOS.[40] Because macOS and iOS share the same kernel, implementing these macOS protections on iOS is technically straightforward.[41]

B. Dr. Rubin Does Not Support His Assertion that Unique Features of Mobile Devices Makes Exclusive Distribution Necessary.

70. While Dr. Rubin cites a number of features that he asserts make mobile devices unique from a security perspective, he does not explain why any of them necessitates exclusive app distribution. For example, Dr. Rubin stresses that there are over one billion active devices and that the App Store hosts two million apps that have been downloaded over 180 billion times. However, that is a retroactive justification. Dr. Rubin does not cite any evidence that Apple knew when it adopted its exclusive distribution model that the platform would achieve this size. And, importantly, Dr. Rubin does not explain why such statistics require all iOS apps to be sold exclusively through Apple's App Store. As I explain below, given the inherent tradeoff between

---

[36] https://www.cloudflare.com/learning/ssl/what-is-an-ssl-certificate/.
[37] [redacted]
[38]
[39]
[40] Federighi Dep. 80:2-5; 77:24-78:13.
[41] PX2888 at ¶125.

accuracy and efficiency, the large volume of apps on the App Store makes it *harder* for a centralized review program to reliably catch all malicious submissions.

71. Similarly, Dr. Rubin argues that the iOS threat model should reflect the growing sophistication of malicious actors in building malicious apps for mobile devices. Dr. Rubin, however, does not compare or distinguish the sophistication of attackers that target mobile devices with those that target personal computers. Dr. Rubin states that social engineering attacks are on the rise, but he does not explain why this would distinguish the threat model on mobile devices from that on personal computers.

72. Dr. Rubin claims without evidence that users store more private information on mobile devices because of their portability and convenience. The storage of sensitive information is not unique to mobile devices. In fact, personal computers may contain even more sensitive data than iOS devices because users and entities often do not store historical, legal, or financial documents on their phones (*e.g.*, banking documents, leases, wills, tax documents, business documents) but they may store such documents on their personal computers.

73. Moreover, information on mobile devices is frequently transferred to personal computers when users sync data across devices. Indeed, synchronizing data across devices is not just something that is technically possible but is something that Apple encourages.[42] As such, private and sensitive data from iPhones, including credit card information, passwords and contacts, is often stored on Mac devices via iCloud.

74. Finally, Dr. Rubin also claims that iOS devices are small and portable and therefore might be misplaced, lost or stolen at a higher frequency than other computer systems. But, again, he fails to explain what relationship this has to Apple's requirement that all iOS apps be sold through the Apple App Store. The security controls that protect lost and stolen devices are completely orthogonal to app distribution and would continue to protect iOS devices, even if iOS were to implement a more open distribution model. For example, the iOS operating system protects data on a lost or stolen device with Touch ID, Face ID, and complex passcodes. iOS also includes a security feature called Find My iPhone that enables device owners to remotely erase all data from their devices if they are lost or stolen.[43] Not only do these security features protect sensitive user data when a device is lost or stolen, but they also deter thieves and other bad actors.[44]

VII. Dr. Rubin's Comparison of Security Between Platforms is Misleading and Incomplete.

75. Dr. Rubin compares security on iOS to a variety of other platforms—Android, Windows, and Linux—that implement a more open distribution model than iOS. These comparisons are misleading and incomplete. Notably missing from Dr. Rubin's analysis is Apple's macOS—a platform designed and operated by the same company based on the same kernel with much of the

[42] *See, e.g.*, Overview to syncing your Mac and your devices - Apple Support.
[43] *Id.*
[44] https://www.lifewire.com/security-settings-iphone-thieves-hate-2487730.

same user-base as iOS, but which allows for third-party distribution. Apple believes that macOS is secure despite offering a more open distribution channel, stating that "apps from both the [Mac] App Store and the internet can be installed worry-free" [45], and that "macOS layers protections to ensure that apps downloaded from the internet are free of known malware".[46]

A. Dr. Rubin's Quantitative Analysis is Flawed.

76. Dr. Rubin relies on unreliable and misleading statistics that overstate the security of iOS relative to Android. In particular, Dr. Rubin relies on the database of Common Vulnerabilities and Exposures ("CVEs"), which contains only "publicly disclosed cybersecurity vulnerabilities."[47] Because the database is limited to "publicly disclosed" vulnerabilities, it likely undercounts vulnerabilities on closed platforms, such as iOS. The list of CVEs on iOS is significantly limited by Apple's decision not to "disclose, discuss, or confirm security issues until an investigation has occurred and patches or releases are available."[48] Further, even external security researchers need Apple's permission when disclosing iOS vulnerabilities, which illustrates Apple's control over how many CVEs could be reported concerning iOS.[49] Dr. Rubin has not considered the fact that Android's open-source nature makes it easier for members of the public to identify and report CVEs. As a result of these limitations, CVEs can be biased by differences in platform openness and disclosure policies and cannot reliably be used as a measure of relative security between the two platforms.

77. Dr. Rubin references several studies that suggest that Android is less secure than iOS, but these studies are also unreliable because 1) the comparisons can only involve identified malware and 2) there are many reasons why platforms may differ in amount of malware beyond the platforms' security capabilities (*e.g.*, demographics or susceptibility of users). It is very difficult to design a comprehensive study to compare the security of different platforms. For example, the Nokia threat intelligence report that Dr. Rubin references to suggest that Android devices are fifty times more likely to be infected with malware than iOS devices does not appear to control for differences in market share among Android and iOS devices. While the statistics show that Android comprises a greater overall percentage of infected devices than iOS, these figures are also driven by each platform's volume share of devices. Even if studies show that Android contains more malware than iOS, in practice, it is very difficult to isolate the effect to non-exclusive distribution.

B. Dr. Rubin Does Not Control for Differences Unrelated to App Distribution Model.

78. There are fundamental differences between the iOS and Android platforms, which make it improper to wholly attribute differences in security to the app distribution models of the two

[45] https://www.apple.com/macos/security/.

[46] https://manuals.info.apple.com/MANUALS/1000/MA1902/en_US/apple-platform-security-guide.pdf.

[47] https://cve.mitre.org/cve/.

[48] https://support.apple.com/en-us/HT212146.

[49] PX2934.

platforms. First and foremost, as Dr. Rubin acknowledges, the iOS platform is much more uniform than Android because device makers can customize their own versions of the Android operating system. These device makers must then release their own customized upgrades and bug patches, which often happen at infrequent intervals, if at all. This fragmentation is material because whereas on iOS "the bugs are getting addressed, and fairly quickly[,] Android bugs can persist for months if not years, and there's a pretty good chance that a given device will just be abandoned after two or three years without even security updates."[50] As such, attackers can target older versions of Android with known vulnerabilities. This issue has nothing to do with Android's app distribution model, and opening third-party distribution channels on iOS would not prejudice Apple's ability to automatically roll out security patches and operating system updates.

79. Android and iOS are also differentiated by their developer identity verification and code signing mechanisms. For instance, Android allows developers to self-sign their apps using certificates that can be generated without the assistance of trusted certificate authorities. Android also does not require developers to verify their identities prior to distributing apps to the device. However, opening distribution on iOS would not necessarily require Apple to allow developers to self-sign their apps, nor would it require Apple to forgo identity verification of developers. As I described previously in my testimony, third-party distribution channels could rely on trusted certificate authorities to verify developers' identities and issue certificates.

C. The Android Platform in China is a Less Relevant Comparison than the macOS Platform in the United States.

80. Dr. Rubin presents a "parade of horribles" argument, suggesting that iOS would resemble the Android platform in China if iOS were to permit third-party app distribution. I disagree with that comparison. *First*, as noted above, there are unique features of the Android security model that are entirely distinct from iOS. Apple does not have to adopt those features in order to allow for third-party distribution. *Second*, Dr. Rubin's "case study" of Android in China tacitly attributes all of the security issues he describes to the app distribution channel. But he shows no evidence justifying that attribution and fails to consider other factors affecting that ecosystem—including a much more fragmented Android ecosystem, much cheaper hardware, less sophisticated device makers and other factors. Dr. Rubin's suggestion that enabling third-party distribution on iOS would lead to an ecosystem that mirrors Android in China in this respect is unsubstantiated. Dr. Rubin has performed no formal analysis or comparison that would allow him to make any meaningful conclusions based on the Android market in China.

VIII. Dr. Rubin Significantly Overstates the Security Benefits of App Review

81. Dr. Rubin asserts that App Review is a "critical component" of the iOS security architecture. Dr. Rubin significantly overstates the security benefits of App Review.

[50] https://venturebeat.com/2019/10/17/as-ios-13-hits-50-adoption-android-fragmentation-keeps-getting-worse/.

A. App Review Is Limited in Its Ability to Catch Sophisticated Security Attacks.

82. App Review can help filter out apps that contain obvious security violations; however its ability to detect sophisticated security attacks is limited.

83. Apple's App Review process is subject to the well-recognized tension between accuracy and efficiency; this tradeoff is exacerbated by the fact that Apple acts as the sole gatekeeper for apps distributed on the iOS platform. The App Store processes roughly 100,000 iOS apps per week, spanning all categories of apps.[51] To accommodate this demand, Apple employed over 430 reviewers as of July 2020.[52] App Reviewers typically review between 50 to 100 apps per day and productivity is internally tracked.[53] The review process typically takes three to seven minutes, and not more than 13 minutes for new apps;[54,55] in some instances, App Reviewers took as little as 32 seconds to approve two apps.[56]

84. Apple appears to prioritize efficiency over accuracy in this tradeoff. In an internal email, Mr. Haun states: "There are iOS apps that are very deep, and we don't get to all of the parts of them. That's OK and should be the norm. We review what we can quickly and competently get to . . . we will never have all the expertise or time to go into everything."[57] Eric Friedman, head of Apple's Fraud Engineering Algorithms and Risk Unit at Apple, has also acknowledged in an internal email that App Review does not "accomplish anything that would deter a sophisticated attacker" and should be regarded "as little more than the equivalent of the TSA at the airport".[58] Thus, while Apple could spend more time screening for security violations in apps, particularly malicious apps that require additional time to evaluate due to the level of sophistication involved, it is currently unable to due to the sheer volume of apps that must be reviewed. Even if Apple were able to spend more time reviewing each app, it would still struggle to detect sophisticated attacks—bad actors are constantly innovating new techniques to obfuscate malware. Third-party reviewer channels on iOS could focus on a narrower band of apps, allowing each reviewer to specialize in and spend more time per app.

85. The limitations of Apple's App Review process are exemplified by public accounts of the malicious apps that have slipped through Apple's review process over time. One such example is the Jekyll app, which I mentioned earlier in my testimony. After Apple was informed of the Jekyll vulnerability in 2013, Apple internally acknowledged that Jekyll "[was] a huge exposure" and that "there's clearly a lot that [had] to be done in App Review to deal with this".[59]

---

[51] PX2337.76.
[52] PX314.21.
[53] PX6.2.
[54] Shoemaker Dep. 56:15-17.
[55] Haun Dep. 276:22-25.
[56] PX131.1.
[57] PX140.1.
[58] PX251.1.
[59] PX2080.1.

86. My Jekyll paper is not the only documented instance of malware evading detection during App Review. For example, Apple's App Review process failed to detect thousands of third-party apps that covertly downloaded private user data, such as email addresses and device identifiers, using obfuscated private APIs in a third-party advertising SDK called YouMi. Apple was seemingly unaware of the data breach until it was reported by a third-party code analytics platform, SourceDNA, in 2015. In response to SourceDNA's report, Apple investigated the data breach claims and found that YouMi's usage of private APIs was written in a way that was difficult for certain App Review tools (*i.e.*, static analyzers) to detect.[60] As a result, App Review had improperly approved 3,825 applications that harvested user data.[61]

87. Malicious apps continue to evade detection during App Review. In 2020, third-party security researchers discovered that more than 20 malicious apps using similar methods as described in my Jekyll paper remained in the iOS App Store.[62] In 2019, a security firm called Wandera identified 17 apps on the App Store that fraudulently collected ad revenue through a Trojan attack.[63] Specifically, this attack allowed apps to continuously open web pages in the background or click links without any user interaction.[64] Apple removed the apps from the App Store as soon as Wandera reported its findings. However, the developers of one of the malicious apps, "EMI Calculator & Loan Planner", resubmitted the app for review without removing the malicious behavior; Apple approved it again for the App Store, despite its prior removal.[65] In response to this mistake, Mr. Kosmynka acknowledged that "we are making critical errors" during App Review.[66]

88. Thus, while App Review may be helpful in screening apps for obvious security violations, the process has been and continues to be limited in its ability to detect less obvious, more sophisticated security violations inherent in malicious apps.

B. App Review Screens for Many Non-Security Issues.

89. Furthermore, based on my review of internal Apple data and documents and deposition testimony, I conclude that much of Apple's App Review process is designed to screen for quality assurance, content moderation and commercial compliance issues that are not security related.

90. For quality assurance, App Reviewers are supposed to check for compliance with Apple's human interface guidelines and proper functionality which require apps to adhere to certain themes (*i.e.*, clarity, deference and depth) and design principles (*e.g.*, aesthetic integrity, consistency).[67] Apple also screens for proper functionality, such as checking to see that apps do

[60] PX31.2.
[61] *Id.*
[62] https://blog.zimperium.com/dr-jekyll-and-mr-hide-how-covert-malware-made-it-into-apples-app-store/.
[63] https://www.wandera.com/ios-trojan-malware/.
[64] *Id.*
[65] PX2084.1.
[66] *Id.*
[67] PX9.14; https://developer.apple.com/design/human-interface-guidelines/.

not crash or have a bug that "significantly hinders the usability of the application".[68] These are not inherently security properties.

91. App Review is also supposed to evaluate each app submission for objectionable content and intellectual property usage.[69] Again, these are not security properties.

92. While Dr. Rubin claims that objectionable content can be associated with security vulnerabilities, he does not explain why such security vulnerabilities would not be addressed by Apple's security-related screening functions (including its dynamic and static analyzers). For example, he claims that over 25% of attacks on mobile devices come from pornography-related malware. While certain malware attacks might attempt to use pornography as a lure to get consumers to download malicious software, that does not mean that pornography itself is an inherent malware risk; iOS already has security protections in place to detect and screen for malware, regardless of whether the app is associated with objectionable content.

93. App Review further screens for "business model appropriateness".[70] This screening involves, among other things, ensuring that the price meets certain criteria and that the app does not use an unauthorized payment system for in-app purchases.[71] None of Apple's criteria for commercial compliance affect iOS security.

94. Most App Review rejections are for non-security reasons. Apple publishes an internal weekly report of App Review metrics that show the App Review process rejects apps for security reasons relatively infrequently. For example, for the week ending March 7, 2020, of the top 10 reasons listed for apps rejected during App Review, only one category—"Legal: Data collection & storage"—could directly implicate a security concern, depending on the context.[72] The 3,490 apps that violated this property represent only a fraction of the total number of app rejections.[73] My evaluation of data points from other historical periods yields similar results, with only a fraction of apps rejected for security-related reasons.

[68] PX101.155.
[69] PX9.20.
[70] PX9.25.
[71] PX101.71; https://developer.apple.com/app-store/review/guidelines/.
[72] PX2337.76.
[73] *Id.*

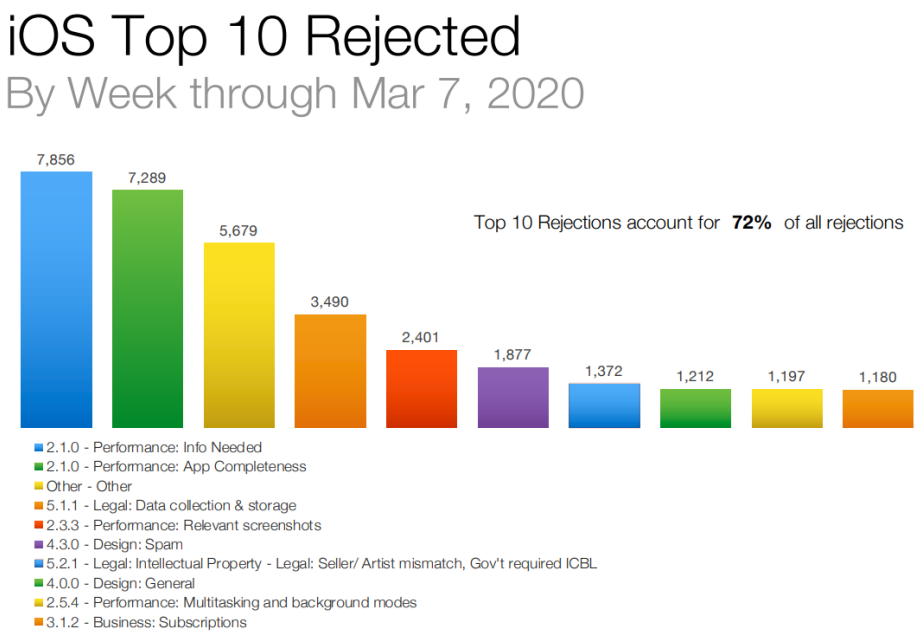

Figure 1: PX2337.76.

95. With respect to App Review's non-security screening steps, my view is that content moderation and quality assurance testing are common and regularly performed by many organizations using a variety of techniques. For instance, members of Internet Association, a group of 43 leading internet companies, "use a variety of techniques—including human review and machine learning—to evaluate posts, images, videos, and other content, and restrict or remove access to content that is illegal or otherwise violates their community standards".[74] Content moderation and quality assurance testing are also common features of technology platforms that enable third-party applications or content. For example, Amazon publishes and enforces quality and content criteria for apps or games offered on the Amazon Appstore.

IX. Eliminating Exclusive App Distribution on iOS Would Not Lead to an Experience that Is Any Less Secure than What Apple Already Accepts on macOS.

96. Dr. Rubin claims that centralizing the distribution of apps and prohibiting the distribution of apps outside of the App Store allows Apple to add a protective layer for iOS devices. He further claims that permitting alternative third-party app stores would increase instabilities and security risks for iOS. I disagree for the reasons that follow.

A. Users Could Choose to Exclusively Download Apps through Apple's App Store.

97. Permitting third-party distribution channels on iOS would have no impact on the security of the iOS App Store or users' ability to download apps from the iOS App Store. Users could choose to download apps exclusively through the iOS App Store in an open distribution model—retaining all of the same security features identified by Dr. Rubin in his testimony. Mr.

[74] https://internetassociation.org/positions/content-moderation/.

Federighi confirms that "if a user [chooses] to purchase from—only the App Store, the user would have the same protections that the user enjoys today".[75]

B. Apple's On-Device Security Features Are Designed to Ensure that the Impact of Malware Downloaded onto an iOS Device Remains Localized.

98. A more open distribution model on iOS would provide users choice in deciding the level of security they prefer in an app distribution channel. For example, users could choose to download apps that have not been signed or screened for security issues and instead rely on the on-device protections. In choosing to do so, each user would accept some risk.

99. I disagree with Dr. Rubin's assertion that iOS platform security "is only as strong as its weakest link". Instead, there is redundancy in the system. Apple's on-device security features are designed to ensure that the impact of malware downloaded onto an iOS device remains localized. [redacted] In other words, iOS is designed to prevent viruses, a certain type of malware that aims to infect other files. This goal is largely implemented by mandatory sandboxing of all third-party apps. As Mr. Shoemaker acknowledged during his deposition, "while no amount of review would…stop a virus from getting into the system…the sandbox model certainly disallows a little application from bouncing around the system".[77]

100. In addition to sandboxing third-party apps, the on-device security feature XN prevents a virus from executing any additional code outside of the app binary and from injecting other apps with malware.

101. Because of these on-device security protections, Mr. Shoemaker agrees that "to the extent anyone claims that Apple needs a close guard on iOS [distribution] to prevent viruses, for instance, that is just not true".[78] As such, even if a user chose to download an app that had not been signed or screened for security issues, the app would not reduce the security of *other* apps on the user's device or the security of the overall iOS ecosystem.

C. Third-Party App Stores Can and Do Provide Secure Distribution Channels.

102. I disagree with Dr. Rubin's assertion that third-party app stores are not incentivized to provide a secure experience. Dr. Rubin is not an economist and provides no basis for his opinion that the "incentives" that apply to the App Store would not apply to any other company in the market. However, I note that there are numerous examples of third-party app stores that invest in security.

[75] Federighi Dep. 87:17-21.
[76] [redacted]
[77] Shoemaker Dep. 486:1-20.
[78] *Id*. 488:12-16.

103. For example, Aptoide is a third-party app store on Android that advertises security as its top priority.[79] It claims to accomplish this through an app review process that includes both a human and an automated review, which is comprised of six different anti-viruses.[80] Once an app has been uploaded to the store, "apps are rescanned over and over to ensure that no malware is missed" and "user input is then used by the security team to update the anti-malware system".[81] As a result of these efforts, a 2017 academic study showed that Aptoide was the safest Android marketplace, safer than even the Google Play Store.[82] Such empirical evidence suggests that third parties indeed have incentives to keep their stores secure, and may even do a better job than the platform operator.

104. Dr. Rubin also assumes that third-party app stores will lack the resources to review apps in the way that Apple does. Dr. Rubin's argument assumes, without support, that every third-party app store would be forced to contend with the same sort of volume the App Store does. Some third parties may choose to run an app store that specializes in narrower bands of apps, which would reduce the volume of apps that the third party would be required to screen. Operating at a smaller scale than Apple, and the roughly 100,000 apps its App Review processes each week, would allow third parties to achieve the same level of security as Apple's App Store, while requiring fewer resources than Apple.

D. Implementing On-Device Malware Scanning Would Not Weaken iOS Security.

105. Dr. Rubin suggests that unlike macOS, iOS is ill-positioned to allow third-party malware scanners because of its mandatory sandboxing. He claims that malware apps "would need operating system hooks to perform real time monitoring". This is misleading. The Darwin kernel, which is shared by macOS and iOS, already contains these "hooks" that antivirus software could utilize to monitor system events for signs of malware. With only a few modifications to the sandboxing profile, third-party antivirus tools could receive the information they need for analysis and continue to run in a user-level sandbox that restricts their ability to access or modify other files on the device. This is a well-known technique and would not weaken iOS security.

106. Further, Apple could introduce its own malware scanning tools (i.e., XProtect), which it has already built and deployed on macOS. Apple representatives testified that it would be technically feasible to do this. This would not degrade iOS security because it would not require Apple to provide third parties with any additional privileges.

E. Enabling Third-Party Distribution Is Not Equivalent to "Jailbreaking."

107. Dr. Rubin's comparison of a macOS-like app distribution model (*i.e.*, allowing third-party distribution) to jailbroken iOS devices is flawed and misleading. While jailbreaking indeed

[79] https://en.aptoide.com/company.

[80] https://blog.aptoide.com/is-it-safe-to-use-aptoide/, https://blog.aptoide.com/evolution-of-aptoide-malware-detection-system/.

[81] https://blog.aptoide.com/is-it-safe-to-use-aptoide/.

[82] https://nsl.cs.waseda.ac.jp/wp-content/uploads/2018/04/submitted_wama2017.pdf.

facilitates the installation of unauthorized software outside of the App Store, it is a user-initiated exploit that bypasses many of the *on-device* security mechanisms that are present on iOS, such as sandboxing. Unlike on jailbroken devices, on-device security mechanisms on non-jailbroken devices will continue to protect users, even if Apple allows third-party distribution channels.

X. Open Distribution Can Improve Security on iOS

108. As noted above in paragraphs 83 and 84, the App Review process suffers from the tension between accuracy and efficiency. Apple's App Review team reviews 100,000 apps per week. Indeed, Apple representatives testified that processing time is a point of emphasis for the App Review team; Apple strives to review 50% of apps within 24 hours of submission and 90% of apps within 48 hours of submission.[83] The emphasis on turnaround time often means that App Reviewers work long hours and are put under pressure to increase their output.

109. If permitted on iOS, third-party distribution channels could implement review processes that are less burdened by the volume of apps distributed on their respective platforms. Third parties could also develop alternative approaches to screening for security issues, some of which might improve upon Apple's current approach. For instance, a third party might create an app store that focuses on a narrower category of apps. This focus could allow specialized app reviewers to spend more time with each app in both manual and automated review, resulting in the reviewers gaining a deeper understanding of the security issues most relevant to each app category.

110. Specialization in specific categories of apps could also allow the reviewers to focus on and develop novel tools to screen for security concerns. For example, an app store that specializes in distributing business apps (*e.g.*, office management, inventory management, HR, finance applications) might focus on screening for and developing tools to detect security issues that are unique to this category, such as secure connections to enterprise networks. A transition from a "one-size-fits-all" review model could enhance security for the broader iOS ecosystem.

111. Lastly, Dr. Rubin appears to assume that if there were additional distribution channels and associated review processes on iOS that they would be completely siloed and would not communicate with one another. I do not believe this assumption is supported. In my experience, the security industry is constantly evolving through collaboration and developing on existing techniques.

XI. The Security and Privacy Benefits Offered by Apple's IAP Can Also Be Offered by Third Party Payment Solutions.

112. Dr. Rubin claims allowing third-party payment systems could lead to less secure payment mechanisms and differing security standards that would facilitate bad acts. However, third-party payment systems can and already do ensure secure and private payment transactions on iOS.

[83] PX6.3.

113. Exclusive use of IAP is not required to provide secure in-app purchases on iOS. When asked the reasons why Apple decided that all digital sales must use IAP, Mr. Forstall replied that it was to "make it easier for developers to sell digital goods"; that it provided "a different mechanism for how to help developers make money"; that it provided a consistent user experience; and that it was "to avoid apps from trying to basically circumvent [the revenue split]".[84] Mr. Forstall did not provide any security justifications for Apple's mandated use IAP for the sale of digital goods.

114. Furthermore, Apple requires iOS developers to use third-party payment settlement providers, like Square or Stripe, when selling physical goods or services through native iOS applications. Apple itself uses third-party payment settlement providers to process its IAP transactions. This suggests that Apple trusts these third-party payment settlement providers to facilitate secure transactions on its platforms.

115. Payment systems that accept, transmit, or store cardholder data are governed by strict industry security standards. Specifically, these systems are required to be compliant with the Payment Card Industry Data Security Standard ("PCI DSS").[85] The PCI DSS requires various protective measures including firewall installation and maintenance, encrypted transmission of cardholder data, use of anti-virus, as well as the regular testing of security systems and processes amongst other measures. Table 1 outlines the goals and requirements of the PCI Data Security Standards; this demonstrates that many of Apple's security practices that Dr. Rubin describes (*e.g.*, restricted access to customer data, system data collection and monitoring) are industry norms and not exclusively practiced by Apple.

[84] Forstall Dep. 252:14-254:10.

[85] "PCI FAQs." PCI Compliance Guide. https://www.pcicomplianceguide.org/faq.

Table 1: The PCI Data Security Standard (2018)[86]

Goals	PCI DSS Requirements
Build and Maintain a Secure Network and Systems	1. Install and maintain a firewall configuration to protect cardholder data 2. Do not use vendor-supplied defaults for system passwords and other security parameters
Protect Cardholder Data	3. Protect stored cardholder data 4. Encrypt transmission of cardholder data across open, public networks
Maintain a Vulnerability Management Program	5. Protect all systems against malware and regularly update antivirus software or programs 6. Develop and maintain secure systems and applications
Implement Strong Access Control Measures	7. Restrict access to cardholder data by business need to know 8. Identify and authenticate access to system components 9. Restrict physical access to cardholder data
Regularly Monitor and Test Networks	10. Track and monitor all access to network resources and cardholder data 11. Regularly test security systems and processes
Maintain an Information Security Policy	12. Maintain a policy that addresses information security for all personnel

116. Third party payment processors like Square and Stripe can and do provide secure alternatives to Apple's IAP system—they are both Level 1 PCI DSS compliant. As such, Square and Stripe are required to adhere to the strictest security standards and are subject to audits of their security systems. Further, because these third-party payment processors are already compatible with iOS applications and utilized by iOS developers, they can perform the front-end

[86] "PCI DSS Quick Reference Guide." PCI Security Standards Council. https://www.pcisecuritystandards.org/documents/PCI_DSS-QRG-v3_2_1.pdf.

functions that Apple's IAP system currently performs for purchases of digital products, just as they do for physical goods and services.

117. Dr. Rubin states that fraud detection techniques become more effective with larger quantities of data. He then uses this claim to argue that by requiring all developers to use IAP for all sales of digital goods, Apple can aggregate more transaction data, and that this data could allow Apple to improve the accuracy of its fraud algorithms. However, Dr. Rubin fails to consider the fact that the data Apple acquires through IAP is much narrower in scope than that used by large third-party payment systems servicing numerous customers through numerous channels. For instance, in 2019, Apple facilitated approximately $61 billion in sales of digital goods and services.[87] Stripe, however, processes "hundreds of billions of dollars" in transactions each year. This disparity in transaction volumes suggests that third-party processors facilitate more transactions and can thus use this additional data to better tune their fraud detection algorithms.

118. Dr. Rubin claims that another benefit to requiring the use of IAP is Apple's use of a tamper-resistant Hardware Security Module ("HSM") and other mechanisms for safeguarding user data. However, HSMs are not unique to Apple—for example, HSMs have been around for almost half a century. There are several companies that manufacture HSMs, including IBM, D'Crypt Private Limited, Motorola, and Entrust, that can be utilized by third party-payment processors to implement data security.

119. Apple makes available to third-party developers its mechanisms for safeguarding user data, including certain public APIs that ask the user to authenticate transactions using Touch ID or Face ID.[88] [redacted]

[87] https://www.apple.com/newsroom/pdfs/app-store-study-2019.pdf.
[88] https://developer.apple.com/documentation/localauthentication/.
[89] [redacted]

* * *

Pursuant to 28 U.S.C. § 1746, I declare under penalty of perjury that the foregoing is true and correct and that I executed this written direct testimony on April 27, 2021, in Atlanta, Georgia.

WORD COUNT: 10,965

Wenke Lee, Ph.D.